IN THE UNITED STATES DISTRICT COURT
FOR THE SOUTHERN DISTRICT OF NEW YORK

INTERNATIONAL BUSINESS MACHINES CORPORATION, **Plaintiff,** **-vs.-** **PLATFORM SOLUTIONS, INC. and T3 TECHNOLOGIES, INC.,** **Defendants.**	**Civil Action No. 06 CV 13565 (LAK)** **<u>PUBLIC VERSION</u>**

PLAINTIFF IBM'S OPENING CLAIM CONSTRUCTION
<u>BRIEF FOR THE *MARKMAN* HEARING</u>

April 21, 2008

TABLE OF CONTENTS

Page

TABLE OF AUTHORITIES

Page

Cases

Statutes and Other Authorities

Plaintiff International Business Machines Corporation ("IBM") respectfully submits its Opening Claim Construction Brief for the *Markman* hearing.

I. INTRODUCTION

Defendant Platform Solutions, Inc. ("PSI") has tried to build a business based on theft of IBM's patents, trade secrets, and copyrights. PSI makes a knock-off of IBM's mainframe computers. PSI's product is an "emulator" – something that mimics or imitates a different system. [redacted]

[redacted]

[redacted]

[1] "Ex. __" refers to the indicated Exhibit of the Declaration of Edward DeFranco.

This action involves ten IBM computer technology patents that are infringed by PSI's emulator. The patents fall into five general areas of computer technology:[2]

- Instruction/Architecture Patents: U.S. Patent Nos. 5,987,495 ("the '495 patent") and 6,775,789 ("the '789 patent").
- Emulation Patents: U.S. Patent Nos. 5,953,520 ("the '520 patent") and 6,009,261 ("the '261 patent).
- Floating Point Patents: U.S. Patent Nos. 5,687,106 (the '106 patent), 5,696,709 ("the '709 patent"), and 5,825,678 ("the '678 patent").
- Partitioning Patents: U.S. Patent Nos. 6,654,812 ("the '812 patent") and 6,971,002 ("the '002 patent").
- I/O Patent: U.S. Patent No. 5,414,851 ("the '851 patent").

Pursuant to this Court's scheduling order, IBM and PSI have had extensive discussions concerning claim terms/phrases contained in these patents. The terms discussed in this brief were selected through negotiation by IBM and PSI and are listed in the Joint Claim Construction And Prehearing Statement that was filed with the Court on March 28, 2008.[3] PSI's proposed constructions for several important terms, including "processor," "register," "instruction," and "converter," all deviate from the plain and ordinary meaning of those terms, as PSI itself has used them, and attempt to add unsupported limitations to avoid infringement.

IBM believes that most of the disputed claim terms should be given their plain and ordinary meaning, and virtually all of the terms discussed in this brief are under consideration now because PSI has proposed constructions that seek to artificially narrow and limit IBM's patent claims. Having chosen to implement its business model and attempting to mimic IBM's mainframe computers without the benefit of a patent license, PSI now seeks to have this Court

---

[2] The patents-in-suit are attached as Exhibits 13-22 of the DeFranco Declaration. To simplify the citations, they are referred to herein by the last three digits of the patent number, along with the cited column and line numbers.

[3] IBM has reduced the number of asserted claims to simplify the issues in this case. The asserted claims are 4, 7, 9, 11, 12, 19, and 22 of the '495 patent; claims 1, 4, 8, 13, 15, 32, and 33 of the '789 patent; claims 1, 2, 7-12, and 15 of the '261 patent; claims 1, 2, 4, 9, 10, and 12 of the '520 patent; claims 3, 5, and 7 of the '709 patent; claims 1, 6, 7, 9, and 12 of the '678 patent;
(footnote continued)

read limitations into the asserted claims that are not supported by the intrinsic or extrinsic evidence.

II. BACKGROUND OF THE PARTIES

For over forty years, IBM has invested massive amounts of time, effort, know-how, creativity, and money into developing and improving its mainframe computer architectures and the computer systems that implement those architectures. As a result of its investments over time, IBM has developed System z. System z is the brand name for IBM's current mainframe computer systems. System z evolved from IBM computer systems dating back to 1964. Its predecessors include System/390® ("S/390®"), which was introduced in 1990. System z is an umbrella term for: zSeries® servers, the z/OS operating system, and other IBM software that runs on zSeries® servers in conjunction with the z/OS operating system.

PSI was formed in 1999 as a spin-off from Amdahl Corporation, a company that used to make – pursuant to a patent cross-license with IBM – mainframe computers that were compatible with IBM mainframes. PSI seeks to move – or "migrate" – customers from IBM mainframes to PSI's emulator platforms, which are based on Intel Itanium® processors.

PSI's sales pitch to customers includes assertions that a PSI emulator will "transform" an Intel-based computer into a system that will run IBM's mainframe operating systems – including z/OS – and other copyrighted IBM software, and will act as if it were an IBM computer system. PSI's emulators infringe IBM patents – a fact that has been confirmed by IBM's analysis of a PSI emulator, the source code for the PSI emulators, and other materials produced by PSI in this litigation. Analysis of PSI's technology has confirmed broad and systematic infringement of many IBM patents, including patents covering important aspects of IBM's computer architectures as well as the very emulation techniques that PSI has used to attempt to mimic those architectures.

---

claims 11, 12, 13, 15, and 17 of the '106 patent; claims 1 and 11 of the '812 patent; claims 1, 9, and 17 of the '002 patent; and claims 9, 21, 22, 29, and 30 of the '851 patent.

III. OVERVIEW OF THE TECHNOLOGY

To better understand the asserted patents and the terms/phrases in dispute, we will first describe IBM's mainframe computer architectures and systems, and PSI's infringing system. We will incorporate into that discussion a very general discussion of the subject matter of the asserted patents and PSI's infringement.

A. IBM's Mainframe Computer Architecture And Systems

In very basic terms, a mainframe computer is a large computer capable of processing large amounts of data with a high degree of accuracy, reliability and security. All computers can be defined by their architecture, much like a building is defined by a blueprint. A computer's architecture is more complex because it defines the interrelationship between the computer system and the software (including application programs and operating systems) that runs on it. The central component of a computer architecture is the specification of all of the instructions that the computer can perform. Each instruction represents a basic function, *e.g.*, an arithmetic operation such as addition or an input/output operation. Programs are composed of many of these instructions. Because each computer architecture defines its own instruction set (including a particular formatting and layout for the instructions), under normal circumstances a particular piece of software will, without modification, only run correctly on the particular architecture for which it was produced.

In the 1960's IBM pioneered with its System/360 product line the idea of a complete line or "family" of computers that used multiple different hardware implementations, but where each member of that family of computers supported the same architecture. At a development cost of more than $5 billion, System/360 was the "largest privately financed commercial project ever undertaken." (Ex. 23.) Because of the common architecture, a single piece of software could run on any computer in the System/360 line, a profound innovation that significantly reduced the cost of software development and has been called one of "the three most important business innovations of all time." (Ex. 24.) System z and the z/Architecture at issue in this case are the

product of more than forty years of continuing innovation and evolution, and many more billions of dollars of investment, building on IBM's ground-breaking System/360 foundation.

IBM's current mainframe computer systems implement IBM's z/Architecture, which includes almost all of the facilities of IBM's earlier Enterprise Systems Architecture/390 ("ESA/390") as well as a number of new facilities. Much of this architecture is described in a document that IBM makes available to the public, the *z/Architecture Principles of Operation.* (*z/Architecture Principles of Operation* (7th Ed. February, 2008) (available at http://publibz.boulder.ibm.com/epubs/pdf/dz9zr006.pdf).) This document enables third-party vendors to freely create new software systems that work with IBM's z/Series servers. Many instructions of the z/Architecture represent inventive features resulting from IBM's investment, and several of the asserted patents in this case cover aspects of the functionality of these instructions.

Applications written for the z/Architecture interact with and depend upon operating systems that are designed and written to run on computers that implement the z/Architecture and thus capitalize on the features and characteristics of that architecture. Operating systems comprise the fundamental software that controls the execution of programs on the computer and provides basic services such as resource allocation, workload flow, scheduling, input/output control, and data management. The relationship between IBM's computer architectures and the operating systems designed to run on those architectures is one of the important factors contributing to the accuracy and reliability of IBM's computer systems, to customer acceptance of those systems, and to the ability of IBM's computer systems to compete with alternative systems offered by IBM's many competitors. IBM's mainframe operating systems include OS/390, designed to run on computers implementing the ESA/390 Architecture, and z/OS, designed to run on computers implementing the z/Architecture.

Operating systems and application programs are made up of instructions based on the relevant computer architecture. IBM's *z/Architecture Principles of Operation* describes over 500 z/Architecture instructions that may be used by operating systems and applications written to be

compatible with the z/Architecture. Instructions essentially tell the computer system what to do, and the useful functions that computers perform are accomplished by executing millions of instructions per second in order to complete complex tasks. The simplest instructions, such as addition and subtraction, are typically executed directly by the hardware portion of a processor. This provides fast performance but limited flexibility. More complicated instructions can be executed using microcode. Microcode is software that is part of a processor. It provides more flexibility because microcode can be more easily changed during the design process than hardware. Microcode can be described as the "connecting gear" between an instruction and the processor's hardware.

B. PSI's Infringing Emulators

Emulation uses one data processing system to imitate another data processing system, with the goal of allowing programs written for one computer architecture to properly execute on a computer with a different architecture, so that the imitating system essentially accepts the same data, executes the same programs, and achieves the same results as the imitated system. For example, IBM's System z operating system, z/OS, is designed to execute specifically on a data processing system implementing IBM's z/Architecture. Through emulation, however, it is theoretically possible to run the z/OS operating system on a data processing system that implements a completely different architecture.

PSI has developed an emulator for IBM's mainframe computer architectures. PSI claims that its emulator runs the z/OS and OS/390 operating systems on a processor that implements the Intel® Itanium® architecture. PSI also claims that its emulator allows the imitating Itanium® system to accept the same data, execute the same programs, and achieve the same results as the conventional IBM data processing system.

PSI's emulator system adds a layer of software, which PSI refers to as its "firmware," to an Intel-based computer to purportedly allow the Intel computer to understand and execute IBM's OS/390 and z/OS operating systems and related applications. Because the IBM instructions are not compatible with the non-IBM architecture of the Intel-based computers, PSI

claims its layer of software translates each IBM instruction so that it can be understood and executed by the Intel computer. All of the relevant parts of the IBM server and all of the relevant data structures (including all of those recited in the claims of the asserted patents) are necessarily present in PSI's system, either physically or by emulation. This allows the PSI firmware to provide an emulated z/Architecture processor.

Overall, PSI's goal is to enable computer programs based on IBM's z/Architecture and ESA/390 Architecture to run on computer systems that are based on a very different architecture – Intel's Itanium® Architecture – and to allow those Intel-based computers (which otherwise are incapable of understanding and executing IBM instructions) to mimic and appear to function as IBM mainframes. To accomplish this goal, PSI must, among other things:

(i) emulate IBM's z/Architecture and ESA/390 Architecture on the Intel-based computer by translating all of IBM's z/OS and OS/390 architecture instructions into Intel architecture (Itanium®) instructions that an Intel-based computer can understand and execute;

(ii) effect such instruction translation so as to ensure that the resulting Itanium® instructions will perform operations that are performed by IBM mainframes, including, for example, "floating point" operations, in logically the same way those operations are performed by IBM mainframes, so that the Intel-based computer will act like an IBM mainframe, in that programs written for the relevant IBM mainframe architectures will execute properly;

(iii) partition the Intel-based computer system and/or enable communications among partitions on that system in the same way that partitioning is done by IBM so that the Intel-based system will act more like an IBM mainframe and enable users to appear to achieve the same or similar results as achieved on an IBM mainframe; and

(iv) manage input/output functionality in ways that seek to mimic the relevant behavior and functionality of an IBM mainframe.

PSI's so-called "Open Mainframe" systems seek to do all of the above.

[redacted] These include, but are not limited to, the instruction set patents previously mentioned, and the additional IBM patents described below.

IV. THE LAW OF CLAIM CONSTRUCTION

The construction of a patent claim is a question of law for the Court. *Markman v. Westview Instruments, Inc.*, 52 F.3d 967, 979 (Fed. Cir. 1995) (*en banc*), *aff'd*, 517 U.S. 370 (1996). This determination is based primarily on the intrinsic evidence, which includes the claims, the specification, and the prosecution file history. *Markman*, 52 F.3d at 979.

Claim language. "The words of a claim are generally given their ordinary and customary meaning," that is, "the meaning that the term would have to a person of ordinary skill in the art in question at the time of the invention, *i.e.*, as of the effective filing date of the patent application." *Phillips v. AWH Corp.*, 415 F.3d 1303, 1312-13 (Fed. Cir. 2005) (*en banc*) (citations omitted). Further, "the context in which a term is used in the asserted claims can be highly instructive." *Id.* at 1314.

Specification. While the claim language itself is critical in determining a proper construction, a person of ordinary skill in the art would also read the disputed claim(s) in the context of the entire patent, including the specification and prosecution history. *Phillips*, 415 F.3d at 1313; *see also Brookhill-Wilk 1, LLC v. Intuitive Surgical, Inc.*, 334 F.3d 1294, 1298 (Fed. Cir. 2003).

An accused infringer cannot overcome the heavy presumption of ordinary meaning "simply by pointing to the preferred embodiment or other structures or steps disclosed in the specification or prosecution history." *Sunrace Roots Enterprise Co. Ltd. v. Sun Victory Trading Co., Inc.*, 336 F.3d 1298, 1305 (Fed. Cir. 2003) (*quoting CCS Fitness Inc. v. Brunswick Corp.*, 288 F.3d 1359, 1366 (Fed. Cir. 2002)). Indeed, it is inappropriate to read the specification into the claims and thereby narrow them beyond what is provided in the claim language itself. *Intamin Ltd. v. Magnetar Technologies, Corp.*, 483 F.3d 1328, 1335 (Fed. Cir. 2007) ("As this

court has repeatedly noted, a narrow disclosure in the specification does not necessarily limit broader claim language.") (citations omitted); *see also Verizon Services Corp. v. Vonage Holdings Corp.*, 503 F.3d 1295, 1302-03 (Fed. Cir. 2007) ("The mere fact that the specification's examples of translation may involve a change in protocol from a higher to a lower level protocol does not establish that such a limitation should be imported into the claims."). In particular, an accused infringer cannot seek to narrow a patent claim based on an embodiment disclosed in the specification. *Inverness Medical Switzerland GmbH v. Warner Lambert Co.*, 309 F.3d 1373, 1379 (Fed. Cir. 2002) ("It is improper to limit the claim based on a preferred embodiment of the invention.") (citations omitted); *see also Saunders Group, Inc. v. Comfortrac, Inc.*, 492 F.3d 1326, 1332 (Fed. Cir. 2007) ("A patent that describes only a single embodiment is not necessarily limited to that embodiment.")

"An applicant is not required to describe in the specification every conceivable and possible future embodiment of his invention." *Sunrace Roots,* 336 F.3d at 1305 (*quoting Rexnord Corp. v. Laitram Corp.*, 274 F.3d 1336, 1344 (Fed. Cir. 2001)). To overcome the heavy presumption of ordinary meaning an accused infringer must point to things in the specification and the file history that are definitional – as opposed to merely descriptive – of the preferred embodiments, so as to demonstrate "an intent to deviate from the ordinary and accustomed meaning of a claim term by redefining the term or by characterizing the invention in the intrinsic record [*i.e.*, the specification or file history] using words or expressions of manifest exclusion or restriction, representing a clear disavowal of claim scope." *Id.*, 336 F.3d at 1304.

The "heavy presumption" that claim terms have their ordinary meaning may be overcome "where the patentee, acting as his or her own lexicographer, has clearly set forth a definition of the term different from its ordinary and customary meaning." *Brookhill-Wilk,* 334 F.3d at 1298-99.

Prosecution File History. The prosecution file history may illustrate how the inventor and the United States Patent and Trademark Office understood the claim scope, particularly in

light of the prior art. *Phillips*, 415 F.3d at 1317. However, the prosecution file history is less useful in claim construction because it lacks the detail of the specification. *Id.*

Extrinsic Evidence. While extrinsic evidence, such as treatises and dictionaries, may aid in the understanding of the technology of a patent, the specification is the "single best guide to the meaning of a disputed term." *See Phillips*, 415 F.3d at 1314-15 (*quoting Vitronics,* 90 F.3d at 1582). Further, extrinsic evidence cannot be used to contradict the intrinsic evidence to define claim terms. *See Dow Chem. Co. v. Sumitomo Chem. Co.*, 257 F.3d 1364, 1373 (Fed. Cir. 2001).

The Federal Circuit has explained that "[d]ictionaries or comparable sources are often useful to assist in understanding the commonly understood meanings of words and have been used both by our court and the Supreme Court in claim interpretation." *Phillips*, 415 F.3d at 1322. "A dictionary definition has the value of being an unbiased source 'accessible to the public in advance of litigation.'" *Phillips*, 415 F.3d at 1323 (*quoting Vitronics Corp. v. Conceptronic, Inc.,* 90 F3d 1573, 1585 (Fed. Cir. 1996). While a claim construction "should not rise or fall based upon the preferences of a particular dictionary editor," dictionaries are useful to understand the meaning of a term as used in the context of a patent. *Id.* at 1322-24.

> [E]xtrinsic evidence in the form of expert testimony can be useful to a court for a variety of purposes, such as to provide background on the technology at issue, to explain how an invention works, to ensure that the court's understanding of the technical aspects is consistent with that of a person of skill in the art, or to establish that a particular term in the patent or the prior art has a particular meaning in the pertinent field."

Phillips, 415 F.3d at 1318.

Means-Plus-Function. The patent statute provides that "[a]n element in a claim for a combination may be expressed as a means or step for performing a specified function without the recital of structure, material, or acts in support thereof, and such claim shall be construed to cover the corresponding structure, material, or acts described in the specification and equivalents thereof." 35 U.S.C. § 112, ¶ 6 (2007). Such a claim element is referred to as "means-plus-function," and it recites a function to be performed, rather than definite structure. *Signtech USA Ltd. v. Vutek, Inc.*, 174 F.3d 1352, 1355 (Fed. Cir. 1999). To construe a means-plus-function

claim, one first determines the function of the limitation. The next step is to determine the corresponding structures described in the specification for performing that function. *See Kemco Sales, Inc. v. Control Papers Co.*, 208 F.3d 1352, 1360 (Fed. Cir. 2000). The means-plus-function limitation is construed to cover only the corresponding structures and their equivalents.

"A computer-implemented means-plus-function term is limited to the corresponding structure disclosed in the specification and equivalents thereof, and the corresponding structure is the algorithm." *Harris Corp. v. Ericsson, Inc.*, 417 F.3d 1241, 1253 (Fed. Cir. 2005); *see also WMS Gaming Inc. v. Int'l Game Tech.*, 184 F.3d 1339, 1349 (Fed. Cir. 1999) ("In a means-plus-function claim in which the disclosed structure is a computer, or microprocessor, programmed to carry out an algorithm, the disclosed structure is not the general purpose computer, but rather the special purpose computer programmed to perform the disclosed algorithm."); *Aristocrat Technologies Australia Pty Ltd. v. Int'l Game Tech.*, 2008 WL 819764, *4 (Fed. Cir. March 28, 2008) ("In cases involving a computer-implemented invention in which the inventor has invoked means-plus-function claiming, this court has consistently required that the structure disclosed in the specification be more than simply a general purpose computer or microprocessor."). In other words, one looks to the specification for the algorithm to perform the recited function.

Importantly, "[t]he term 'algorithm' is not limited to a formula of mathematical symbols" or "specific source code for the computer." *Computer Acceleration Corp. v. Microsoft Corp.*, 516 F.Supp. 2d 752, 763 (E.D. Tex. May 7, 2007). Rather, a patentee might express "the steps, formula or procedures to be performed by the computer", *i.e.* the algorithms, "textually, or shown in a flow chart." *Id.* (*citing Application of Freeman*, 573 F.2d 1237, 1245-46 (C.C.P.A. 1978) and cases cited therein). Regardless of the format, the structure need only be described such that "one of ordinary skill in the art can determine the limitations on what is claimed." *Id.* (*citing Budde v. Harley-Davidson, Inc.*, 250 F.3d 1369. 1381-82 (Fed. Cir. 2001); *In re Dossel*, 115 F.3d 942, 946-47 (Fed. Cir. 1997)).

V. FOUR DISPUTED CLAIM TERMS COMMON TO SEVERAL PATENTS

IBM addresses below the patents at issue by technical area. Because four of the disputed claim terms appear in several patents, and because both parties have proposed competing constructions for these terms that apply to all of the patents, IBM will begin by addressing these four terms. IBM will then address the remaining disputed terms on a patent-by-patent basis. These common terms are:

(1) "processor" ('495, '789, '520, '261, and '709 patents),

(2) "instruction" ('495, '789, and '709 patents),

(3) "program status word" ('495 and '678 patents), and

(4) "register" ('495 and '789 patents).

IBM will address each one of these four terms in turn.

A. "Processor"[4]

The primary dispute between the parties is whether a processor is limited to "one or more integrated circuits" or whether, as is consistent with the plain and ordinary meaning and with the preferred embodiments, a processor may include software that is not part of an integrated circuit. The parties' respective construction are as follows:

IBM's CONSTRUCTION	PSI's CONSTRUCTION
A portion of a computer system that interprets and executes instructions	One or more integrated circuits that process coded instructions and perform a task

"Processor" appears in claims of the '495, '789, '520, '261, and '709 patents. These claims do not limit the term "processor" to "one or more integrated circuits" as PSI proposes.

[4] The term "processor" is used in claims 4, 7, 19 of the '495 patent; claims 1, 15, 32, and 33 of the '789 patent; claims 1, 9, 10, and 12 of the '520 patent; claims 3, 5, and 7 of the '709 patent; and claims 1, 7, 9-12, and 15 of the '261 patent.

- Claims 1, 7, and 19 of the '495 patent recite a method of (or apparatus for) "operating [a] processor," with no additional limitations on the processor element.
- Claims 1 and 33 of the '789 patent recite "one or more processors of the computing environment."
- Claims 1 and 9 of the '520 patent recite a processor (or a method of operating a processor) "which has a native instruction set and emulates instructions in a guest instruction set."
- Claims 1 and 7 of the '261 patent recite "an emulation method for executing an incompatible computer program on a target processor."
- Claims 3, 5, and 7 of the '709 patent recite "a processor that executes a floating point instruction."

Because patent claim terms deserve their broadest reasonable interpretation, it would be improper to narrowly limit the term "processor" to mean "one or more integrated circuits." *Rexnord Corp. v. Laitram Corp.*, 274 F.3d 1336, 1342 (Fed. Cir. 2001) ("In addition, unless compelled to do otherwise, a court will give a claim term the full range of its ordinary meaning as understood by an artisan of ordinary skill."); *see also Free Motion Fitness, Inc. v. Cybex Intern, Inc.*, 423 F.3d 1343, 1348-49 (Fed. Cir. 2005).

IBM's construction is supported by the specifications of the patents, none of which limit a processor to "one or more integrated circuits." In fact, the phrase "integrated circuit" is not found anywhere in the specifications of the '495, '789, '709, or '261 patents. Thus, contrary to PSI's position, the specifications clearly establish that software is a component of a processor and can perform a key role in executing the instructions.

<u>The '495 Patent.</u> The '495 specification describes the use of microcode (*i.e.*, software) to implement the handling of instructions in the CPU:

> CPU 102, which constitutes the primary instruction processing unit of system 100, may comprise one or more central processors (CPs) (not separately shown). As is conventional in the art, CPU 102 has an instruction decoder for decoding instructions being executed as well as an execution unit for executing the decoded instructions. *These may be implemented by any suitable combination of hardware and microcode in a manner well know in the art.*

('495 patent, col. 7:21-28, emphasis added.) The '495 specification plainly teaches that a processor is not limited to one or more integrated circuits and can be implemented by hardware and microcode (*i.e.*, software). While PSI may argue that this hardware and microcode must be implemented by "one or more integrated circuits," there is no such limitation in the '495 claims, specification, or prosecution history. Further, the above portion of the '495 specification teaches that a processor decodes or "interprets" instructions and "executes" those decoded instructions. This language directly mirrors IBM's proposed construction, which is "a portion of a computer system that interprets and executes instructions." The PSI construction, which excludes the possibility of using microcode, reads out the preferred embodiment of the '495 patent.

<u>The '789 Patent.</u> Figures 11-13 and column 14, lines 19-60 of the '789 patent describe central processing complexes according to the invention and illustrate such complexes having multiple processors, each with its own processor identifier and timing facilities. The specification does not describe the processors being implemented using one or more integrated circuits. The Central Processing Units ("CPUs") identified in Figs. 1 and 3 are also not limited or discussed in terms of "integrated circuits." Again, the specification clearly establishes that software is a component of a processor and performs a key role in executing the instructions. The '789 specification makes clear that it uses microcode to execute an instruction:

> As is known, the instruction is slowed down by having the microcode wait to issue an endop, which would then return control to the next instruction.

('789 patent, col. 13:28-31.) Thus, PSI's construction reads out the preferred embodiment of the '789 patent because it would exclude the use of microcode. That is incorrect.

<u>The '520 Patent.</u> The '520 patent discloses a technique for adapting a processor with particular software to allow it to interpret and execute a new "guest" instruction set:

> When implemented as a PowerPC processor, each CPU 4 preferably *comprises* a single integrated circuit superscalar microprocessor. . . Each CPU is *further adapted* in accordance with the present invention to execute guest instructions (e.g., CISC instructions or some other instruction set that is not native to CPU 4) by emulation. As described further hereinbelow, guest instructions 20 are each emulated by fetching and executing one or more semantic routines 19, which each contain two or more native instructions.

('520 patent, col. 4:19-45, emphasis added.) The specification makes it clear that the hardware element "microprocessor" is combined with the "semantic routines" (*i.e.*, software) to comprise the processor. Without the semantic routines that enable the processor to perform the emulation, the guest instructions could not be interpreted and executed. PSI's construction would read out the preferred embodiment of the '520 patent because it would exclude the use of semantic routines.

The '261 Patent. The '261 patent discloses the use of software within a processor to emulate instructions from a different architecture:

> The preferred embodiment described in detail herein uses a *microcoded* implementation *in the target processor. The microcode is software* which is protected from being changed by user software executing in the target processor.

('261 patent, col. 8:61-65, emphasis added.) The '261 specification also recites that the "target processor also *contains* preprocessing function 102 (*id.*, col. 8:39-40, emphasis added) that may be "implemented in microcode, or simply be performed by a program executing on the target processor." (*Id.*, col. 4:15-19.)

These quotations from the specification make it clear that the '261 patent considers the use of software in the processor to interpret and execute an additional set of instructions to be a key element of the invention. In contrast, when the specification wishes to particularly point out the hardware component of the processor, it does so explicitly such as in the references to "target processor hardware" ('261 patent, col. 12:4) and "processor chip" (*Id.,* col. 13:57.) PSI's construction would again read out the '261 preferred embodiment because it would, for example, exclude the use of microcode.

Other Intrinsic Evidence. The discussion of the CPU in the *IBM Principles of Operation* (intrinsic evidence because it is incorporated by reference in the '789, '709, and '495 patents)[5] makes clear that it is the logical function of the processor (rather than its physical

---

[5] *See* '789 patent, col. 5:11-14; '709 patent, col. 1:15-17; '495 patent, col. 2:2-4.

implementation) that is relevant: "The central processing unit (CPU) is the controlling center of the system. It contains the sequencing and processing facilities for instruction execution, . . . The physical implementation of the CPU may differ among models, but the logical function remains the same." (Ex. 25.)

Extrinsic Evidence. IBM's proposed construction is supported by the most relevant extrinsic evidence. The primary definition for "processor" found in IBM's *Dictionary of Computing* does not limit the definition of a processor to one or more integrated circuits: "In a computer, a functional unit that interprets and executes instructions." (Ex. 26, *IBM Dictionary of Computing*, 10th Ed., p. 533, (1994).) In turn, a "functional unit" is defined as: "An entity of hardware or software, or both, capable of accomplishing a specified purpose." (Ex. 27, *IBM Dictionary of Computing*, 10th Ed., p. 292, (1994).) In addition, a person of ordinary skill in the art at the time of the invention would understand that the term processor could include more than just "one or more integrated circuits." (*See* Smotherman Decl. ¶¶ 5-8.[6]) While PSI will undoubtedly cite dictionary definitions and other extrinsic evidence that describe a processor as "one or more integrated circuits," the terms of a patent's claim must be construed with the broadest reasonable scope, which encompasses but is not limited to "one or more integrated circuits."[7]

In light of the plain and ordinary meaning of the term, the unrestricted use of the term in the claims, the use of software in processors in the preferred embodiments, and the extrinsic evidence, a processor is correctly construed as "a portion of a computer system that interprets and executes instructions."

[6] Dr. Smotherman's Declaration is attached at Tab C.

[7] PSI relies on the narrower dictionary definition [2] from the *IBM Dictionary of Computing*, while IBM relies on definition [1].

B. "Instruction"[8]

IBM's CONSTRUCTION	PSI's CONSTRUCTION
A language construct that specifies an operation and identifies its operands, if any.	*A string of digits* that specifies an operation and identifies its operands, if any, *and can be directly executed by the processor to which it is directed.*

The term "instruction" was well-known to a person of ordinary skill in the art at the time of the invention, and has the plain and ordinary meaning proposed by IBM. Apart from the use of the term in the claims, the term is not defined in the specifications or prosecution histories of the patents that use the term (the '495, '789, '709 and '261 patents). As a result, the parties are relying mainly on extrinsic evidence.

IBM's use of the term "language construct" is based on the definition of "instruction" in the *IBM Dictionary of Computing*: "A language construct that specifies an operation and identifies its operands, if any." (Ex. 28, *IBM Dictionary of Computing*, 10th Edition, p. 346 (1994).) PSI's proposed phase, "a string of digits," is narrower in that it is limited only to "digits." This narrow construction has no intrinsic support. Moreover, as discussed in Section VIII below, PSI's phrase is applicable to a specific type of an instruction, a "machine instruction," and not the broader general term "instruction." (*See* Smotherman Decl. ¶¶ 9-12.) For example, an *instruction* set forth by a programmer (*e.g.*, assembly code) could read "ADD A,B" whereas the corresponding *machine instruction* for that operation (*e.g.*, compiled machine code) consists of the string of representative ones and zeros. *Id.*

In addition, the limiting phrase PSI adds at the end of its construction for instruction – "can be directly executed by the processor to which it is directed" – is ambiguous and has no intrinsic or extrinsic support. PSI presumably adds this phrase to make a noninfringement

[8] The term "instruction" appears in claims 1-3, 5-11, 14-15 and 18-21 of the '495 patent; claims 4, 8, 13, 15, 23 and 31 of the '789 patent; and claims 3, 5 and 7 of the '709 patent. It also (footnote continued)

argument. PSI's Itanium® processor cannot execute an IBM instruction until it is translated by PSI's firmware. If it obtains its construction, PSI will likely argue that there is no infringement because the IBM instruction is something different than that which is directly executed by the Itanium® processor in PSI's system. This only further highlights why PSI's construction is too limiting under the proper analysis. Moreover, there is nothing in the intrinsic evidence which limits the term "instruction" in this way.

PSI's proposed construction should be rejected on its face because rather than provide greater clarity, which is the goal of the claim construction process, it only creates additional confusion. *See Fieldturf USA v. Sports Constr. Group,* 499 F. Supp. 2d 907, 926 (D. Ohio 2007) (refusing to accept a proposed claim construction because "[a]dding this to the claim construction would not serve to help the jury but would instead cause confusion.") The Court should reject PSI's construction for that reason alone.

Extrinsic Evidence. Among all of the extrinsic evidence PSI cites in the Joint Statement, the evidence identifying "direct" execution is the definition of "machine instruction" from the *IBM Dictionary of Computing*: "An instruction that can be directly executed by a processor of a computer." (Ex. 29, *IBM Dictionary of Computing*, 10th Edition, p. 408 (1994).) Even so, this definition does not support PSI's construction. First, this definition is not for "instruction" but rather for a different term, "machine instruction." Second, this dictionary definition states that the machine instruction "can be directly executed by a processor of *a* computer," and does not limit execution of the instruction as suggested by PSI to "*the* processor *to which it is directed.*" In other words, the dictionary definition simply sets forth that a machine instruction can be directly executed by "a processor," which would include either the IBM processor or the PSI processor – regardless of which processor the instruction was originally written for or "directed to."

---

appears in claims 1, 2 and 7-12 of the '261 patents as part of the terms "incompatible instruction" and "target instruction."

Thus, PSI's attempt to confect a noninfringement position is not supported by the intrinsic or extrinsic evidence, and is also disallowed by the rules of claim construction. *See Phillips,* 415 F.3d at 1315 ("claims 'must be read in view of the specification, of which they are a part'") (*quoting Markman,* 52 F3d at 979.)

C. "Program Status Word"[9]

Both parties agree that a "program status word" is data. The parties' only dispute is whether a program status word is limited to the "contents of the register" (PSI) or "a defined set of data" (IBM):

IBM's CONSTRUCTION	PSI's CONSTRUCTION
A defined set of data that indicates the next instruction to be executed and includes the program condition code and program authority, where that data directs the processor in the execution of a program.	The contents of the register that indicates the next instruction to be executed, includes the program condition code and program authority, and directs the processor in the execution of a program.

The '495 and '678 claims, specification, and extrinsic evidence make clear that a program status word is not limited to PSI's proposed "contents of a register."

The '495 Patent. Independent claims 1 and 14 of the '495 patent recite that a program status word is contained (or located) in a "storage location," a term that can certainly include registers but is broader than merely registers. Similarly, independent claims 7 and 19 of the '495 patent recite that a program status word is contained in a "save area," which is a region in main storage and different from a register. The '495 inventors make clear that a program status word is considered separately from registers: "a processor is controlled by a program status word and by a set of registers defining a program context." ('495 patent, col. 12:6-8.) While the inventors chose not to limit the storage location of a program status word to a register, they did recite registers in other claims. (*See, e.g.*, '495 patent, col. 11:41-42, "said program instruction contains a field specifying a register"; col. 11:44-45, "determining said storage location using the contents

of the Register"; col. 12:12-13, "decoding a program instruction specifying a register selected from said set of registers".) Thus, the inventors could have limited the storage location of a program status word to a register, but chose not to do so, and the construction of that term should not be so limiting.

The '495 specification also never limits the storage location of the program status word to a register. Indeed, it specifically discusses registers within its description of program status word, demonstrating that the patent recognizes a distinction between a program status word and a register. For example, the '495 specification states the following about the program status word and other registers:

> Associated with CPU 102 are a set of 16 32-bit general registers 110 (GR0-GR15), 16 32-bit access registers 112 (AR0-AR15), and a 64-bit program status word (PSW) 114. . . . PSW 114 stores the address of the next instruction to be executed, along with other pertinent state information, such as a condition code and various settable program modes, as described below.

('495 patent, col. 7:42-52.) The inventors again could have limited the program status word to a register like the access registers and general registers, but they chose not to do so. Thus, it is clear from the '495 intrinsic evidence that a program status word is not limited to the "contents of the register."

PSI will argue that the following sentence from the '495 specification implies that the program status word is stored in a register:

> In addition to registers 110 and 112 and PSW 114, CPU 102 has other registers (such as control registers and floating-point registers) that are not relevant to the present invention and are hence not shown.

('495 patent, col. 7:52-56.) To the contrary, the second appearance of the word "and" in the sentence distinguishes registers 110 and 112 from PSW 114. The fact that the sentence goes on to state that CPU 102 "has other registers" does not indicate otherwise because it is referring to registers other than 110 and 112. Moreover, even if the sentence reads as PSI may suggest (*i.e.*,

[9] "Program status word" is used in claims 1, 6, 7, 12, 14, 19, and 22 of the '495 patent, and claims 1 and 7 of the '678 patent.

that the PSW 114 is referred to as a register), the specification is clear elsewhere that the PSW can be stored in different storage locations, consistent with the claim language. "The real address of the PSW 126 in the save area 108 is calculated . . ." ('495 patent, col. 10:63-64.) Because the save area is a separate location from a register, it is clear that the '495 specification is not limiting storage of the PSW (or program status word) to a register. IBM's construction allows for the possibility that a program status word can be stored in a register, but does not restrict it to a register only. PSI's construction, on the other hand, plainly ignores the '495 intrinsic evidence and would read out the preferred embodiment.

The '678 Patent. The '678 claims recite "setting a condition code in a program status word" (*see, e.g.*, '678 patent, claims 1 and 7), but do not limit the location of the program status word to a register, as does PSI's proposed construction.

The '678 specification also does not limit the program status word to the "contents of a register." Figures 2 and 3 of the '678 specification each illustrate a program status word, and the '678 specification indicates that for those illustrations the program storage word is stored in a register. ('678 patent, col. 2:42-43; 2:60-61.) For example, the specification states that "Instruction unit 200 fetches instructions from common main storage 110 according to an instruction address located in the *program status word (PSW) register* 202. . .." (*Id.*, col. 2:40-43, emphasis added.) If "program status word" already meant the contents of a particular register, as PSI proposes, then the phrase "program status word register" would be completely redundant. Thus, this shows that the inventor did not intend "program status word" to be limited to the contents of a particular register, and for this reason alone, PSI's proposed construction should be rejected.

In addition, neither the claims nor the specification of the '678 patent expressly limit the storage location of the program status word. For example, neither indicate that a program status word stored in a memory cache would not be covered by the invention. The examples illustrated in the '678 specification are just that (*i.e.*, examples), and the '678 specification does not state that the program status word must be stored in a register. Further, the '678 specification makes

clear that these descriptions in the specification refer to preferred embodiments, and are not to be used as limitations to the claims:

> While the invention has been described in its preferred embodiments, it is to be understood that the words which have been used are words of description, rather than limitation, and that changes may be made within the purview of the appended claims without departing from the true scope and spirit of the invention in its broader aspects.

('678 patent, col. 4:54-59.) *See Pfizer, Inc. v. Ranbaxy Labs., Ltd.,* 457 F.3d 1284, 1290 (Fed. Cir. 2006) ("But here, the specification ... states that '[t]hese examples are illustrative and are not to be read as limiting the scope of the invention as it is defined by the appended claims.'") PSI's construction impermissibly attempts to read a limitation into the claims from the specification. Because the patentee did not clearly intend for the claims and the embodiments in the specification to be coextensive, PSI's construction must be rejected. *See JVW Enterprises, Inc. v. Interact Accessories, Inc.*, 424 F.3d 1324, 1335 (Fed. Cir. 2005) ("We do not import limitations into claims from examples or embodiments appearing only in a patent's written description, even when a specification describes very specific embodiments of the invention or even describes only a single embodiment, unless the specification makes clear that 'the patentee ... intends for the claims and the embodiments in the specification to be strictly coextensive.'") (citations omitted).

Both parties cite the definition of program status word from the *IBM Dictionary of Computing*, although it does not limit a program status word to the contents of a register, but simply "an area in storage," which is consistent with IBM's construction. (Ex. 30, *IBM Dictionary of Computing*, 10th Ed., p. 539 (1994).) IBM's construction is also supported by *The IEEE Standard Dictionary of Electrical and Electronics Terms*, which similarly does not limit the term to the contents of a register: "(A) A computer word that contains information specifying the current status of a computer program. The information may include error indicators, the address of the next instruction to be executed, currently enabled interrupts, and so

on." (Ex. 31, *The IEEE Standard Dictionary of Electrical and Electronics Terms*, p. 827 (1996).) Finally, a person of ordinary skill in the art at the time of the inventions would have understood that a program status word could be stored in a register or some other type of storage location, such as RAM. (*See* Smotherman Decl. ¶¶ 13-14.)

PSI's construction is unnecessarily narrow, unsupported by intrinsic and extrinsic evidence, and would read out the preferred embodiment of the '495 patent.

D. "Register"[10]

IBM's CONSTRUCTION	PSI's CONSTRUCTION
A part of internal storage having a specified storage capacity and usually intended for a specific purpose.	A hardware storage element in the processor that can be accessed faster than memory.

The term "register" was well-known to a person of ordinary skill in the art at the time of the invention, and has a plain and ordinary meaning reflected in IBM's proposed construction. The claims, specifications, and prosecution histories of both the '495 and '789 patents do not define or limit the term. The parties are consequently relying predominantly on extrinsic evidence to construe the term. IBM's proposed construction is taken verbatim from the *IBM Dictionary of Computing*: "[a] part of internal storage having a specified storage capacity and usually intended for a specific purpose." (Ex. 32, *IBM Dictionary of Computing*, 10th Ed., p. 566 (1994)).

PSI's construction narrows the term "register" to a "hardware storage element in the processor" instead of accepting the broader definition of being part of "internal storage". Neither the claims nor the specification for either the '495 or '789 patents require that a register must be hardware in the processor, and a person of ordinary skill in the art at the time of the invention

[10] The term "register" appears in claims 2-4, 7, 10-11, 15-17, and 19 of the '495 patent, and claims 13 and 31 of the '789 patent.

would have known that a register could be implemented with a combination of hardware and software. (*See* Smotherman Decl. ¶ 15.)

The second part of PSI's construction, namely that a register "can be accessed faster than memory," is similarly inappropriately narrow. Speed of access is simply not an issue that is relevant to these patents, as the '495 and '789 patents make no reference to accessing information in registers faster than information elsewhere. While registers might be accessed faster in many particular implementations, faster access is not a defining characteristic of the term "register" and is not true of all registers. PSI has found one dictionary definition that says that a register is faster than other memory but most definitions, including two more listed on PSI's chart of extrinsic evidence and the *IBM Dictionary of Computing* definition referenced above, make no mention of any speed requirement for a register. Accordingly, IBM is entitled to the broader definition of the term.

The preferred embodiments described in both the '495 and '789 specifications are based on an IBM S/390 environment, in which a register is typically a hardware component in a processor. However, these are simply the preferred embodiments of those patents, and both specifications expressly disclaim that their respective inventions should be limited in that way. (*See* '495 patent, col. 7:7-9; '789 patent, col. 15:12-18.)

IBM's proposed construction is consistent with the plain and ordinary meaning of the term "register," as well as the intrinsic and extrinsic evidence, and should be adopted.

VI. <u>THE INSTRUCTION/ARCHITECTURE PATENTS</u>

A. The '495 "Resume Program" Patent

The '495 patent is directed to the "Resume Program" instruction defined by IBM's z/Architecture. This Resume Program instruction restores the parameters of a program after an interruption has occurred so that normal program execution can be resumed. The claims of the '495 patent are directed to the format and operation of this particular instruction.

The parties dispute the meaning of the four claim terms discussed above: "processor," "instruction," "register" and "program status word." IBM will not repeat the discussion of those

terms here. The parties also dispute the identification of structure in the specification of the '495 patent that corresponds to certain limitations of claim 19 that are in means-plus-function format.

1. "means for decoding a program instruction specifying a register selected from said set of registers, said register pointing to a save area containing a saved program status word and saved register contents" (claim 19)[11]

The parties agree this is a means-plus-function element under 35 U.S.C. § 112, ¶ 6.

IBM's CONSTRUCTION	PSI's CONSTRUCTION
Function: Decoding a program instruction specifying a register selected from the set of registers, the register pointing to a save area containing a saved program status word and saved register contents Structure: Hardware, software, or any suitable combination of the two (*see*, *e.g.*, Fig. 1; 102 5:4-11; or 7:21-28) configured to decode an instruction from a program executing in said problem state specifying a storage location containing a saved program status word (*see*, *e.g.*, Fig. 2A, 2B, 8:63-65, 9:5-15), and equivalents thereof.	Function: Decoding a program instruction specifying a register selected from said set of registers, said register pointing to a save area containing a saved program status word and saved register contents Structure: No corresponding structure.

Function. The parties agree the function for this element is set forth in the claim. For the analysis of all means-plus-function elements in this brief, IBM only discusses the function aspects where the parties materially dispute the function at issue.

Structure. The '495 specification discloses the structure to perform this computer-implemented function of "decoding a program instruction specifying a register selected from said set of registers . . .."

The '495 specification describes (and illustrates in Fig. 1) a CPU (or processor).

[11] The parenthetical that appears after the claim term headings identifies all of the patent claims in which the disputed term appears.

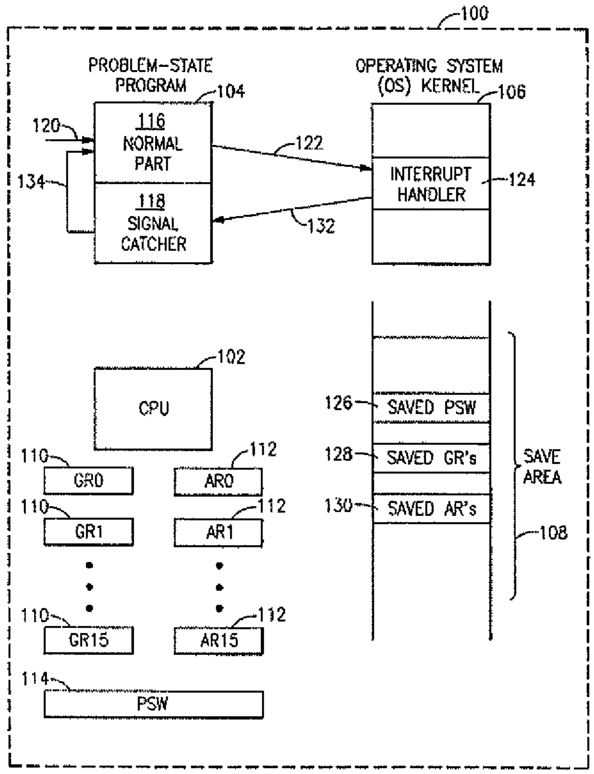

FIG.1

The '495 specification states that "[a]s is conventional in the art, CPU 102 has an *instruction decoder for decoding instructions being executed* as well as an execution unit for executing the decoded instructions. These may be implemented by any suitable combination of hardware and microcode in a manner well known in the art." ('495 patent, col. 7:23-28, emphasis added; *see also*, 5:4-11.) Thus, such a combination of hardware and software performs the function of decoding the instruction.

The '495 specification also provides details about the relevant program instruction, and an algorithm for decoding that instruction to determine the save area. Fig. 2A illustrates the program instruction, including that "a register specification (GR) 204 specifies a general register 110 (GRx) that contains the base address 206 of the save area 108 . . .." ('495 patent, col. 8:63-65.)

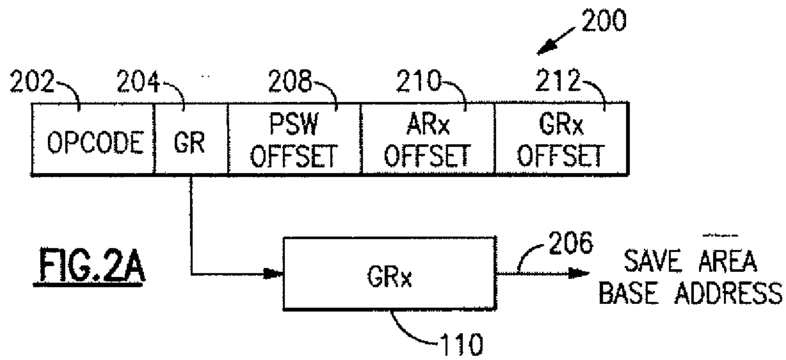

FIG.2A

Fig 2B and col. 9:5-15 of the '495 specification describe the algorithm for decoding this instruction. Thus, the '495 specification discloses structure for decoding the instruction and the algorithm for decoding that instruction. Accordingly, IBM's proposed structure is consistent with the intrinsic evidence, corresponds to the recited function, and should be adopted.

PSI argues that there is no corresponding structure for this claim element. It is well-settled that structure is corresponding if "the specification or prosecution history clearly links or associates that structure to the function recited in the claim." *B. Braun Med., Inc. v. Abbott Labs.,* 124 F.3d 1419, 1424-25 (Fed. Cir. 1997). For limitations involving computer-implemented inventions, the corresponding structure includes algorithms, steps, or software that performs the function. *Aristocrat Technologies,* 2008 WL 8197654 at *4. As cited above, there is ample language in the '495 specification "linking or associating" the structure identified by IBM to this claimed function. Accordingly, PSI's assertion that this there is no corresponding structure for this limitation is incorrect.

2. "means response to said decoding means for executing said instruction, said executing means comprising" (claim 19)

The parties agree this is a means-plus-function element governed by 35 U.S.C. § 112, ¶ 6.

IBM's CONSTRUCTION	PSI's CONSTRUCTION
Function: Executing the instruction Structure: Hardware, software, or any suitable combination configured to execute an instruction (Fig. 1, 102; Col. 5:4-11; Col. 6:66	Function: Executing said instruction Structure: No corresponding structure.

– 7:7, or Col. 7:21-4), and equivalents thereof.	

Structure. The '495 specification discloses the structure to perform this computer-implemented function of "executing said instruction." The '495 specification describes (and illustrates in Fig. 1) a CPU (or processor) as shown above. It states that "[a]s is conventional in the art, CPU 102 has an instruction decoder for decoding instructions being executed as well as an *execution unit for executing the decoded instructions*. These may be implemented by any suitable combination of hardware and microcode in a manner well known in the art." ('495 patent, col. 7:21-28, emphasis added; *see also*, 5:4-11.) Thus, such a combination of hardware and software perform the function of executing the instruction.[12]

The '495 specification also provides the algorithm for executing the instruction as described for the "means for accessing..." and "means for restoring..." steps construed below. As indicated in this claim element, this "executing means" comprises both of those means-plus-function elements.

PSI again erroneously asserts that there is no corresponding structure for this element. Once again, as set forth above, PSI is incorrect because there is ample disclosure in the specification linking the structure cited by IBM to this function.

3. "means for accessing said save area using the contents of the register specified by said program instruction" (claim 19)

The parties agree this is a means-plus-function element governed by 35 U.S.C. § 112, ¶ 6.

IBM's CONSTRUCTION	PSI's CONSTRUCTION
Function: Accessing the save area using the contents of the register specified by the program instruction. Structure: Hardware, software, or any suitable combination of the two (Fig. 1; 102; Col. 7:21-28; or 5:4-11) that is configured to access the save area using the contents of the register	Function: Accessing said save area using the contents of the register specified by said program instruction. Structure: No corresponding structure.

[12] This description also applies to the last two means-plus-function elements of claim 19 (discussed below) because this "executing means" comprises both of those elements.

IBM's CONSTRUCTION	PSI's CONSTRUCTION
specified by the program instruction (Fig. 2A; Fig. 2B, Fig. 3A; Fig. 6, steps 601, 602, 604, 606; Col. 8:63-65; Col. 9:5-22; Col. 10:42-45; Col. 10:46-50; Col. 10:54-59; or Col. 10:62-66), and equivalents thereof.	

Structure. As discussed above for the "means response to said decoding means for executing" claim element, the '495 specification discloses a CPU (or processor) for executing the instruction. Based on the "comprising" language within that element, that CPU description applies to this element as well. The specification also describes an algorithm for "accessing the save area" as set forth below.

The first part of "accessing the save area" is to determine the base address of the save area. Figures 2A and 3A and Figure 6, step 601 illustrate how to use the contents of the general register (GR and GRx fields in 2A, B2 field in 3A) to determine the address of the save area. *See also*, '495 patent, col. 8:58-64; 9:28-31; 9:37-43; and 9:41-44.

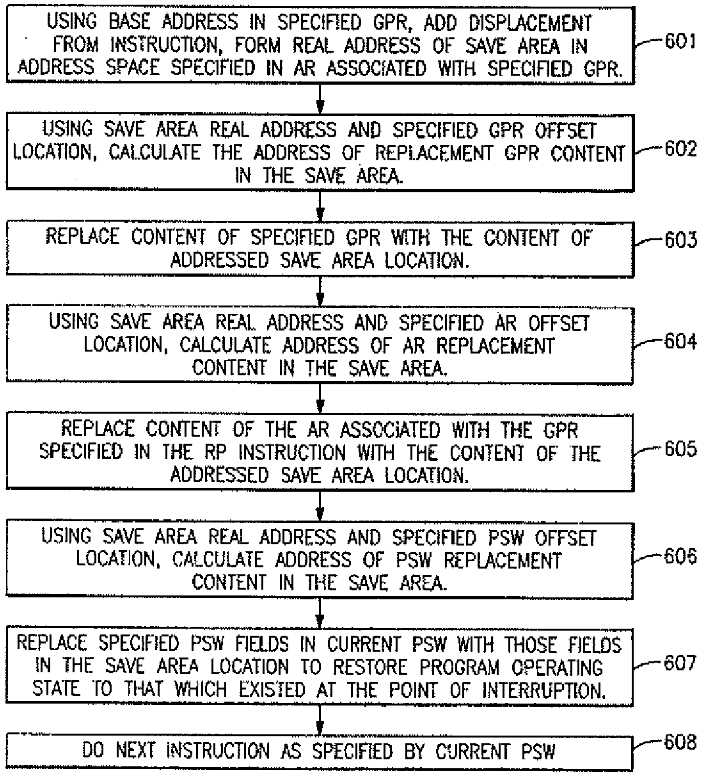

FIG.6

Furthermore, the '495 specification at col. 9:15-22 and col. 9:46-52 describes how to obtain the real address of the save area by converting a virtual address through dynamic address translation. Once the save area address is determined, the save area can be accessed using the save area address and the various offset fields of the Resume Program instruction. Figure 2B and Figure 6, steps 602, 604, and 606 illustrate these steps. The specification also describes the steps of using the save area address combined with the offset fields to calculate the locations of the PSW, general register, and access register within the save area. ('495 patent, col. 9:5-15; col. 10:45-49; col. 10:53-58; and col. 10:61-65.)

PSI again erroneously asserts that there is no corresponding structure for this element. As set forth above, PSI is incorrect because there is ample disclosure in the specification linking the structure cited by IBM to this function.

4. **"means for restoring said program status word and said register from the saved program status word and saved register contents contained in said save area to resume execution at the instruction address contained in said saved program status word with the program context defined by said saved program status word and saved register contents" (claim 19)**

The parties agree this is a means-plus-function element governed by 35 U.S.C. § 112, ¶ 6.

IBM's CONSTRUCTION	PSI's CONSTRUCTION
Function: Restoring the program status word and the register from the saved program status word and saved register contents contained in the save area to resume execution at the instruction address contained in the saved program status word with the program context defined by said saved program status word and saved register contents. Structure: Hardware, software, or any suitable combination of the two (Fig. 1; 102; Col. 7:21-28; or 5:4-11) that is configured to restore the program status word and the register from the saved program status word and saved register contents contained in the same area (Fig. 8, step 810; Fig. 6, steps 603, 605, or 607; Col. 9:58-64; Col. 10:50-53; Col. 10:59-61; or Col. 10:66-11:2), and equivalents thereof.	Function: Restoring said program status word and said register from the saved program status word and saved register contents contained in said save area to resume execution at the instruction address contained in said saved program status word with the program context defined by said saved program status word and saved register contents. Structure: Fig. 6.

Structure. As discussed above for the "means response to said decoding means for executing" claim element, the '495 specification discloses a CPU (or processor) for executing the instruction. Based on the "comprising" language within that element, that CPU description applies to this element as well. The '495 specification also describes an algorithm for this "restoring" step as set forth below.

Steps 603, 605 and 607 in Figure 6 (shown above) illustrate the means for restoring the PSW and registers once their storage locations have been calculated within the save area.

The '495 specification also explains these steps in detail ('495 patent, col. 10:49-52; 10:58-60; and 10:65-11:2) in such a way that a person of ordinary skill in the art could implement the algorithm. *Aristocrat Technologies,* 2008 WL 8197654 at *8 ("the sufficiency of

the disclosure of the algorithmic structure must be judged in light of what one of ordinary skill in the art would understand the disclosure to impart.").

PSI again erroneously asserts that there is no corresponding structure for this element. As set forth above, PSI is incorrect because there is ample disclosure in the specification linking the structure cited by IBM to this function. The '495 specification plainly discloses a CPU or processor for these computer-implemented functions, and a set of algorithms for performing these functions.

B. The '789 "Time Stamp" Patent

The '789 patent covers the implementation of IBM's "Store Clock Extended" instruction. ('789 patent, col. 7:57-61.) As described in IBM's z/Architecture, this instruction directs a processor to store a time-stamp from the processor's clock along with computer system identifier information, and can be used to distinguish transactions or operations of different computer systems based on differences in the time-stamp. *Id.*, col. 1:47-49; 13:54-14:4. The parties dispute the construction of the terms "processor," "instruction," and "register." IBM discusses the meaning of those terms above and will not repeat that discussion here.

The parties also dispute the meaning of the phrase "usable as a current time of day clock value in real-time processing." In addition, the parties dispute the identification of structure in the specification of the '789 patent that corresponds to certain limitations of claim 33 that are in means-plus-function format.

1. "usable as a current time of day clock value in real-time processing" (claims 1 and 33)

IBM's CONSTRUCTION	PSI's CONSTRUCTION
the sequence value reflects the actual time at which the time value was requested, and is usable as a sequential timestamp where later requests always result in larger values	representing the actual time of day at which the value can be used by a program

The '789 patent teaches the generation of "unique sequence values to be used by programs running within the computing environment." ('789 patent, col. 1:23-25.) The claims

of the '789 patent recite generating these "unique sequence values," where each sequence value includes "timing information comprising at least one of time-of-day information and date information," and "selected information usable in making said sequence value unique across a plurality of operating system images on one or more processors of said computing environment . . .." (*See, e.g.*, '789 patent claim 1 (col. 15:21-30) and claim 33 (col. 17:46-18:4).) The claims then recite that this sequence value is "usable as a current time of day clock value in real-time processing by one or more processors of the computing environment," which is the phrase in dispute. (*See, e.g.*, '789 patent, claim 1 (col. 15:28-30) and claim 33 (col. 18:5-7).)

IBM construes this phrase as follows: "the sequence value reflects the actual time at which the time value was requested, and is usable as a sequential timestamp where later requests always result in larger values." This construction is consistent with the '789 intrinsic evidence and should be adopted.

First, the '789 specification provides background about the meaning of "timing information" and a "current time of day clock value":

> Typically, processors of a computing environment either include or have access to timing facilities that provide date and time of day information. In the ESA/390 architecture offered by [IBM], the timing facilities include a time-of-day (TOD) clock, which provides a high-resolution measure of real-time suitable for the indication of the date and time.
>
> In one example, the time-of-day clock is represented as a 64-bit-integer value that is set and *incremented in an architecturally prescribed fashion based on real-time*.

('789 patent, col. 1:27-36, emphasis added.) The first part of IBM's construction, namely that "the sequence value reflects the actual time at which the time value was requested," is consistent with the retrieval of a current time of day clock value as described above. And the second part of IBM's construction, that it is a "sequential value" in which "later requests always result in larger values," is also consistent with an "incremented" time of day clock as described above. In addition, IBM's proposed construction of this claim language as a "timestamp" is supported by the specification of the '789 patent. *See id.*, 1:47-49 and 13:54-14:4.

The '789 prosecution history provides more insight on the disputed phrase, and plainly supports the second part of IBM's construction ("is usable as a sequential timestamp where later requests always result in larger values"). During prosecution of the application that led to the '789 patent, applicants amended claims 1 and 33 to add the phrase "said sequence value representing a *timestamp*" to overcome cited prior art. (Ex. 33, 7/1/02 Response, pp. 21, 28, emphasis added.) In the Remarks section of the Response, applicants described the invention and the basis for adding this phrase to overcome the cited prior art:

> ...in one aspect of applicants' claimed invention, a timestamp is generated that is unique across a plurality of operating system images. This is very different from the teachings of any of the references, either alone or in combination.
>
> For example, while Frey describes a technique for creating a unique value across multiple operating system images, this unique value is not and cannot represent a timestamp. In Frey, information is taken from a timestamp in order to make the unique value; however, the unique value in and of itself is no longer a timestamp. That is, the value generated in Frey is not a timestamp. *There is no discussion at all in Frey of maintaining the generated value as an increasing sequence value, which is needed for a timestamp.*

(Ex. 33, 7/1/02 Response, pp. 16-17, emphasis added.) Applicants later amended these claims again to replace the phrase "said sequence value representing a timestamp" with the phrase in question ("usable as a current time of day clock value in real-time processing by one or more processors of the computing environment"). Applicants again made similar arguments for distinguishing the invention over other prior art ("Wanish"):

> Wanish describes the creation of a unique identifier, as opposed to the creation of a unique current time of day value. The identifier in Wanish includes some time information, but the identifier itself is not *usable as a current time of day value in real-time processing*. This is evident in Wanish by, for instance, the fact that in order to use any time information from the Wanish identifier that information needs to be extracted, since only a portion of the identifier includes the time information, *unlike applicants' sequence value in which the entire value is the timing information.* The extracting of the information takes time, and thus, cannot be used for real-time processing.
>
> * * *
>
> For example, there is no discussion in Wanish of ensuring that the created value is an increasing sequence value, as needed in the time of day value. *Again, while a clock itself produces increasing sequence values, once that clock information is*

> *combined with other information, there is no requirement in Wanish of maintaining the increasing sequence value in the resulting value, so that it can be used for a current time-of-day value.* For instance, assume the technique of Wanish is used to create two ids for two different operating systems. Assume the first id is created on July 8 at 10:00 in the morning and the other is created on July 8 at 10:30 in the morning. To be a time of day value, the second value is to have a greater value than the first. However, there is no such requirement in Wanish, and since information other than time information is also used to create the ids, it is very possible that the ids do not follow in sequence order. Take the following example: using the various information to create the ids, one possible value for the first id is: 123456707081000, and a possible value for the second id is 012345607081030. Thus, while the second id was created after the first, it has a smaller value than the first. Thus, the ids cannot be relied on for current time of day values. There is no requirement in Wanish that the resulting id be usable as a time of day representation. Wanish does not even use the whole clock in its value. Thus, applicants respectfully submit that the ids of Wanish are not usable as time of day values within a computing environment as claimed by applicants, and thus, Wanish does not anticipate applicants' claimed invention.

(Ex. 34, 3/3/04 Preliminary Amendment, pp. 11-12, emphasis added.) The '789 prosecution history plainly defines the disputed phrase as a sequence value that "is usable as a sequential timestamp where later requests always result in larger values."

PSI's construction is not supported by the '789 intrinsic evidence and should be rejected. According to PSI's construction, the sequence value "represent[s] the actual time of day at which the [sequence] value can be used by a program." Nothing in the '789 claims, specification, or prosecution history supports the notion that the sequence value can only be used by a program at the actual time of day reflected in that value. Under PSI's construction, a sequence value with a 10:02AM time value could only be used by a program at 10:02AM. This is nonsensical and completely inconsistent with the '789 intrinsic evidence. It also contradicts the purpose of the invention, which is to time-stamp a record with the then-current time of day and an operating system identifier so that the record has a unique identifier. This unique identifier could not be used to later identify that record if it could only be used at the exact time of day it was created.

PSI cites a number of dictionary definitions and other extrinsic evidence, none of which trump the intrinsic evidence. Thus, IBM's construction should be adopted.

2. **"at least one computer usable medium having computer readable program code means embodied therein for causing the generating of unique sequence values usable within a computing environment, the computer readable program code means in said article of manufacture comprising" (claim 33)**

The parties agree this is a means-plus-function element governed by 35 U.S.C. § 112, ¶ 6 (IBM previously disputed this.)

IBM's CONSTRUCTION	PSI's CONSTRUCTION
Function: Causing the generating of unique sequence values usable within a computing environment. Structure: An article of manufacture (*see, e.g.*, Col. 14:61-15:4) having computer usable media for causing the generating of unique sequence values usable within a computing environment (*see, e.g.*, Fig. 9; Col. 8:62-67; Fig. 10, step 1010, 1002; Col. 9:1-5; Col. 9:13-15; Col. 11:23-35; or Col. 11:61-12:6), and equivalents thereof.	Function: Causing the generating of unique sequence values usable within a computing environment. Structure: No corresponding structure.

Structure. Claim 33 reads as follows:

33. An article of manufacture, comprising:
at least one computer usable medium having computer readable program code means embodied therein for causing the generating of unique sequence values usable within a computing environment, the computer readable program code means in said article of manufacture comprising:
computer readable program means for causing a computer to provide as one part of a sequence value timing information comprising at least one of time-of-day information and date information; and
computer readable program means for causing a computer to include as another part of said sequence value selected information usable in making said sequence value unique across a plurality of operating system images on one or more processors of said computing environment, wherein said sequence value is usable as a current time of day clock value in real-time processing by one or more processors of the computing environment.

The first element of this claim (beginning with the phrase "at least one computer usable medium") identifies the "computer readable program code means," which are then set forth in the second and third elements of the claim. As a result, the structure analysis for this element must

consider the structure set forth in the two referenced elements. Accordingly, IBM will address in Sections (i) and (ii) below the structure in those two referenced elements.

(i) "computer readable program means for causing a computer to provide as one part of a sequence value timing information comprising at least one of time-of-day information and date information" (claim 33)

The parties agree this is a means-plus-function element governed by 35 U.S.C. § 112, ¶ 6 (IBM previously disputed this.)

IBM's CONSTRUCTION	PSI's CONSTRUCTION
Function: Causing a computer to provide as one part of a sequence value timing information comprising at least one time-of-day information and date information. Structure: An article of manufacture (Col. 14:61-15:4) having computer usable media for causing a computer to provide as one part of a sequence value timing information comprising at least one of time-of-day information and date information (Fig. 9; Col. 8:62-67; Fig. 10, step 1002; or Col. 9:1-5), and equivalents thereof.	Function: Causing a computer to provide as one part of a sequence value timing information comprising at least one time-of-day information and date information. Structure: No corresponding structure.

Structure. PSI erroneously asserts that there is no corresponding structure for this element. There is ample disclosure in the specification linking the structure cited by IBM to this function.

The agreed-upon function is performed by the computer readable program code stored on at least one computer usable medium. The '789 specification discloses the structure for this computer usable medium:

> Additionally, at least one program storage device readable by a machine, tangibly embodying at least one program of instructions executable by the machine, to perform the capabilities of the present invention can be provided.

('789 patent, col. 15:1-4.) The '789 specification also discloses an algorithm for this computer readable program code to cause a computer to perform the recited function, as follows:

> Initially, when the STORE CLOCK EXTENDED instruction is executed by a program wishing to obtain the date and/or time of day, an extended time of day clock value is constructed.

('789 patent, col. 8:62-65; Fig. 9, Step 900.) Figure 10 illustrates the logic associated with constructing an extended time-of-day clock value (*id.*, col. 4:57-59) and figure 4 is a representation of such a value. *Id.*, col. 4:38-40. The basic time-of-day (TOD) clock field of the extended TOD clock value (element 404 of figure 4) is a representation of the physical clock used by programs to obtain timing information (*e.g.*, date information and/or time-of-day information). *Id.*, col. 7:4-6. That is, the portion of the extended TOD clock, which is used to obtain timing information for programs, is the basic TOD clock field. In other words, it is the basic TOD clock that "provides as one part of a sequence value timing information comprising at least one of time-of-day information and date information." *Id.*, claim 20. Figure 4 illustrates that the basic TOD clock field is included in bits 8 to s+8 of the extended TOD clock. *Id.*, col. 6:24-25. Step 1002 of figure 10 illustrates how to set the basic TOD clock field. "[B]its 8 to s+8 are set to the value of the running physical clock, STEP 1002." *Id.*, col. 9:2-3. Thus, the structure disclosed to "provide as one part of a sequence value timing information comprising at least one of time-of-day information and date information" is Fig. 9; col. 8:62-67; Fig. 10, step 1002; or col. 9:1-5.

(ii) "computer readable program means for causing a computer to include as another part of said sequence value selected information usable in making said sequence value unique across a plurality of operating system images on one or more processors of said computing environment, wherein said sequence value is usable as a current time of day clock value in real-time processing by one or more processors of the computing environment" (claim 33)

The parties agree this is a means-plus-function element governed by 35 U.S.C. § 112, ¶ 6 (IBM previously disputed this).

IBM's CONSTRUCTION	PSI's CONSTRUCTION
Function: Causing a computer to include as another part of the sequence value selected information usable in making the sequence value unique across a plurality of operating system images on one or more processors of the computing environment. Structure: An article of manufacture (Col. 14:61-15:4) having computer usable media for causing a computer to include as another part of said sequence value selected information usable in making said sequence value unique across a plurality of operation system images on one or more processors of said computing environment (Fig. 9; Col. 8:62-67; Fig. 10, step 1010; Col. 9:13-15; Col. 11:23-35; or Col. 11:61-12:6), and equivalents thereof.	Function: Causing a computer to include as another part of the sequence value selected information usable in making the sequence value unique across a plurality of operating system images on one or more processors of the computing environment, wherein said sequence value is usable as a current time of day clock value in real-time processing by one or more processors of the computing environment. Structure: No corresponding structure.

Function. The parties do not entirely agree on the function for this claim element. IBM contends the function is set forth in the language of the claim and is "causing a computer to include as another part of the sequence value selected information usable in making the sequence value unique across a plurality of operating system images on one or more processors of the computing environment." PSI contends the function includes this phrase, but continues with the claim language "wherein said sequence value is usable as a current time of day clock value in real-time processing by one or more processors of the computing environment." This "wherein" clause is the *result* of the recited function and should not be limited by the 35 U.S.C. § 112, ¶ 6 function/structure analysis. *Lockheed Martin Corp. v. Space Systems/Loral, Inc.*, 324 F.3d 1308, 1319 (Fed. Cir. 2003) ("The function is properly identified as the language after the 'means for' clause and before the 'whereby' clause, because a whereby clause that merely states the result of the limitations in the claim adds nothing to the substance of the claim.").

Structure. PSI again erroneously asserts that there is no corresponding structure for this element. There is ample disclosure in the specification linking the structure cited by IBM to this function.

The recited function is performed by the computer readable program code stored on at least one computer usable medium. The '789 specification discloses the structure for this computer usable medium:

> Additionally, at least one program storage device readable by a machine, tangibly embodying at least one program of instructions executable by the machine, to perform the capabilities of the present invention can be provided.

('789 patent, col. 15:1-4.)

The '789 specification also discloses an algorithm for this computer readable program code to cause a computer to perform the recited function, as follows.

The extended TOD clock as represented in Figure 4 includes a time of day programmable field (TODPF) contained in bits 112-127. ('789 patent, col. 6:28-30). The TODPF "allows the uniqueness of the TOD clock values resulting from representation 400 to be extended to, for instance, different operating system images in a Sysplex environment." *Id.*, col. 8:2-5. Step 1010 of figure 10 illustrates how to include "as another part of the sequence value selected information usable in making the sequence value unique across a plurality of operating system images on one or more processors of the computing environment." "[T]he TODPF field of the TOD register is read and stored into bits 112-127 of the clock representation, STEP 1010." *Id.*, col. 9:13-15.

VII. <u>THE EMULATION PATENTS</u>

The general concept of emulation is discussed above in Section III. IBM is asserting two emulation patents against PSI, the '520 and the '261 patents.

A. The '520 "Address Translation" Patent

The '520 patent discloses a method and system for address translation during emulation of guest instructions in a data processing system. The computer architecture implemented by the system performing the emulation ("the native system") uses one addressing scheme, while the computer architecture implemented by the emulated system ("the guest system") uses another addressing scheme. Each guest instruction to be emulated has a corresponding "semantic

routine" made up of native instructions that facilitates the translation from the guest logical address to the guest real address and thereafter to the native physical address.

The parties dispute the construction of "processor," "instruction set," "semantic routine," and the corresponding structure to certain language found in claim 9. The construction of "processor" is set forth above in Section V.

1. "instruction set" (claims 1, 9)

The parties agree to the construction of "instruction set" with the exception of whether it pertains the operation of instructions of "a computer architecture" (IBM) or "a computer" (PSI).

IBM's CONSTRUCTION	PSI's CONSTRUCTION
The complete set of the operations of the instructions of *a computer architecture* together with the types of meanings that can be attributed to their operands.	The complete set of the operations of the instructions of *a computer* together with a description of the types of meanings that can be attributed to their operands.

IBM's construction is the plain and ordinary meaning of the term in the context of the '520 patent, and is supported by the claim language and the intrinsic evidence offered by both parties. PSI's construction reads out the embodiments of the patent.

The '520 patent relates generally to a method and system for operating a processor that has a "native" instruction set and emulates instructions in a "guest" instruction set. (*See* '520 patent, Claim 1 and 9 Preambles.) The parties have agreed that the term "native" means "pertaining to the *architecture* on which the emulator runs," and "guest" means "pertaining to the *architecture* that is being emulated." (Ex. 35, Exhibit A of the Joint Statement, p.2.) Thus, at the outset, PSI's position contradicts the parties' agreed construction for the terms native and guest. Furthermore, the '520 specification makes plain that both the native and guest instructions are derived from computer architectures (not computers). It explains that, in one embodiment, the emulated guest instruction set is from a CISC *architecture* and the native instruction set is made up of instructions from a RISC *architecture.* ('520 patent, col. 1:45-64.) This demonstrates that